vscode 热部署/热加载/hot deploy/redply　java web项目　　maven+tomcat

目的

自从用过vscode,再也无法忍受eclipse的龟速和庞大的资源占用。但是新入手vsc，还是遇到了诸多困难。
比如热部署。

vsc一切都是基于插件的，所以很强大，但也正因为如此，热部署之类的功能并非开箱即用的。上网搜索了一番，仍然没有找到可用的方案。

要知道，对于ｊａｖａ　ｗｅｂ项目来说，每修改一个文件就需要重新启动ｔｏｍｃａｔ那是何等的痛苦。

折腾过程

准备阶段

vsc,java extention, maven 等

略。

tomcat插件

这个倒是简单，安装tomcat for java这个插件，然后配置插件指向tomcat目录即可搞定（linux系统注意ｔｏｍｃａｔ目录的权限）。
使用方法：

1. 打包war文件：在maven上点clean,然后package即可。
2. 将war传送给tomcat插件。可以在ｗａｒ文件上点右键，run on tomcat; 或者在tomcat插件上点debug war file.

这时候就发现问题了，难道我每修改一行代码都需要package,然后在debug/run on tomat 才可以看到效果？开始新一轮的折腾计划吧…

失败方案

---

tomcat maven plugin

网上有文章说这个插件可以解决热部署，经过实验，这个不行。
1.官网２０１３年后已经停止更新，最新版本是tomcat7-maven-plugin, 不过仍然可以启动tomcat 9.0.
2.这个插件无法和tomcat插件配合，原因在于tomcat插件启动后，是没有manger页面的。而这个插件必须通过manger页面来发布war包。所以只能手动从命令行启动tomcat，然后再通过tomcat7:deploy这个ｍｖｎ命令启动这个插件。虽然这样可以正常工作，但是，看不到ｃｏｎｓｏｌｅ　ｌｏｇ，也无法像网上说的那样通过tomcat7:redeploy命令进行热部署。

各种错误解决：

1. 拒绝连接：原因是tomcat没有启动，这个插件不会自动启动tomcat;

2. 找不到manger页面：原因是tomcat插件启动的ｔｏｍｃａｔ，是单独配置的，这个配置不包含ｍａｎｇｅｒ。所以无法打开localhost:8080/manager/text页面。
3. 授权失败：需要配置manger-ui权限和用户，同时在tomcat和这个插件进行配置。

正确方案

1.修改maven　compile的输出目录

计划把部署目录放到　target/hotsite目录，所以在pom.xml里面做如下修改：

```
<build>
        <!-- 改变ｃｌａｓｓ文件的输出目录 -->
                <outputDirectory>${basedir}/target/hotsite/WEB-INF/classes/</outputDirectory>
                <plugins>
                        <!-- Compile java -->
                        <plugin>
                                <groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-compiler-plugin</artifactId>
                                <version>${maven.compiler.plugin.version}</version>
                                <configuration>
                                        <source>1.8</source>
                                        <target>1.8</target>
                                        <encoding>UTF-8</encoding>
                                </configuration>
                        </plugin>
                        <!-- Build war -->
                        <plugin>
                                <groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-war-plugin</artifactId>
                                <version>${maven.war.plugin.version}</version>
                        </plugin>
</build>
```

2.前端页面的热部署

通过前面的设置ｃｌａｓｓ文件已经完成热部署，但是还有两个部分：一个是ｗｅｂａｐｐ目录下的前端文件，另一个是ｌｉｂ里面的ｊａｒ文件。
有人说如果把class文件输出到 src/main/webapp/WEB-INF/classes　下面，把整个ｗｅｂａｐｐ目录当作部署目录，就不用考虑前端文件了。是这样的。但是，这样做存在很多不好的影响：

1. 源文件和target混淆在一起了。结构混乱。
2. 影响备份。备份源文件一般都是备份src目录，这样一来备份不方便了。
3. 如果原来的src/main/webapp/WEB-INF/lib　里面有文件，就更混乱了。比如部分jar在仓库里面没有，单个项目需要用到，于是手动拷贝过来的。
4. 不知道maven clean的时候是什么结果，反正我是不敢去尝试了。

所以，我的思路是考虑将前端文件热部署过来，到target/hotsite目录。
幸运的是，vsc里面有一个deploy插件。刚好可以完成这个功能。按ctrl+shift+p --> ext install deploy 即可完成安装。然后在vsc的settings.json里面做如下配置：

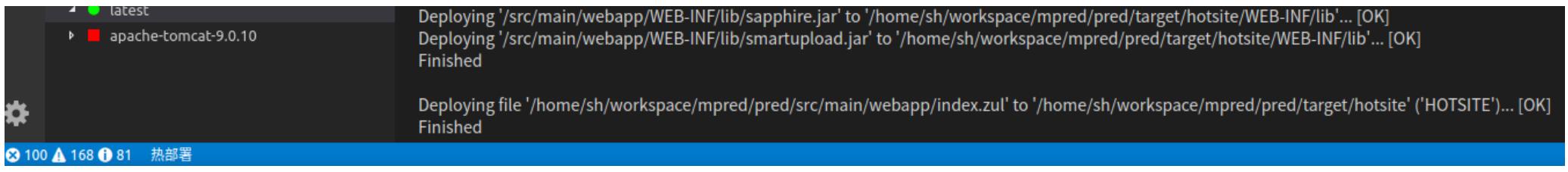
```
"deploy": {
       "packages": [
           {
               "name": "前端",
               "description": "webapp里面的所有文件",
               "files": [
                   "src/main/webapp/*",
                   "src/main/webapp/*.*",
                   "src/main/webapp/*/*.*",
                   "src/main/webapp/*/*/*.*",
               ],
               "exclude": [
                   "src/main/webapp/test/*"
```

```
                ],
                "deployOnSave": true,
                "useTargetList": true,
                "button": {
                    "text": "热部署",
                    "tooltip": "点击这里将前端部署到hotsite",
                    "targets": [ "HOTSITE" ]
                },

            }
        ],
        "targets": [
            {
                "type": "local",
                "name": "HOTSITE",
                "description": "A local folder",
                "dir": "target/hotsite/",
                "mappings": [
                    {
                        "source": "src/main/webapp",
                        "isRegEx": false,

                        "target": "/"
                    }
                ]
            }
        ]
    }
```

配置完成后效果如下图：



下面会出现热部署按钮，点击即可部署整个前端目录到target/hotsite. (提示见上图前 3 行)
当修改某一个网页后按crtl+s保存时，即可自动部署到target/hostsite目录。（提示见上图最后 2 行)

3.jar文件

这个一般不经常改动，直接maven package打包后，复制一份到src/main/webapp/WEB-INF/lib即可。
如果您经常改动，也可以借助ｄｅｐｌｏｙ插件来完成这部分。

4.tomcat虚拟目录

将ｔｏｍｃａｔ的虚拟目录指向target/hotsite，并允许热加载即可。
在tomcat插件上点右键，Open Server Configuration，然后做如下配置：

```
      <Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="false"  >
        <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs" prefix="localhost_access_lo
        <Context path="/p" docBase="ｘｘｘ/target/hotsite" reloadable="true"></Context>
      </Host>
```

重点：
autoDeploy=“false”
reloadable=“true”
docBase=“ｘｘｘ/target/hotsite”

如果您需要重启tomcat，也可以使用autoDeploy=“true”。
ｘｘｘ替换成你自己的项目目录即可。

大功告成

这样终于大功告成了。
第一次，

1. 拷贝lib： maven–>package,然后复制生成的 项目目录或war里面的lib到hotsite;
2. 部署前端页面： 点一下蓝色状态条的 热部署按钮。

以后每次都无需操作了，保存即部署。

实验一下，启动tomcat插件里的tomcat,然后修改一个java或网页文件。无需重启了。

遗憾所在

tomcat控件不支持像添加war包那样添加目录。所以只有借助虚拟目录配置来完成目标。这样的坏处是tomcat插件下面看不到调试的项目，而且docBase=“ｘｘｘ/target/hotsite” 这样等于写死了目录（${workspaceFolder}这样的变量也不支持）。多个项目直接不方便动态启用、停用。